Better Streaming Algorithms for Clustering Problems

Moses Charikar* Princeton University

Liadan O´Callaghan† Stanford University

Rina Panigrahy‡ Cisco Systems

ABSTRACT

We study clustering problems in the streaming model, where the goal is to cluster a set of points by making one pass (or a few passes) over the data using a small amount of storage space. Our main result is a randomized algorithm for the k–Median problem which produces a constant factor approximation in one pass using storage space $O(k\,poly\log n)$. This is a significant improvement of the previous best algorithm which yielded a $2^{O(1/\epsilon)}$ approximation using $O(n^\epsilon)$ space. Next we give a streaming algorithm for the k–Median problem with an arbitrary distance function. We also study algorithms for clustering problems with outliers in the streaming model. Here, we give bicriterion guarantees, producing constant factor approximations by increasing the allowed fraction of outliers slightly.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*computations on discrete structures*

General Terms

Algorithms,Theory

Keywords

Clustering,k-median,streaming algorithm

1. INTRODUCTION

In recent years, there has been a dramatic growth of interest in developing algorithms for massive data sets. In particular, the streaming model has received a lot of attention [3, 27, 18, 13, 22, 16, 15, 14, 1]. Here, an algorithm must process its input by making one pass (or a small number of passes) over it, using a limited amount of memory. This is a common model used for settings where the size of the input far exceeds the size of the main memory available and the only feasible access to the data is by making one or more passes over it.

In this paper, we examine clustering problems in the data-stream model. Clustering problems have been studied extensively across several disciplines. They are typically formulated as optimization problems, where the input is a set of points with a distance function defined on them and the goal is to find a clustering solution (a partion into clusters) that optimizes a certain objective function. Unless otherwise specified, for most of our discussion, we will assume that the distance function is a metric, i.e. is symmetric and satisfies triangle inequality.

A common clustering objective function for is k–Median [17]: find k centers (medians) in a set of n points so as to minimize the sum of distances of points to their closest centers. Recent research has led to the development of very good approximation algorithms for this problem [4, 9, 23, 8, 5]. Another classical clustering objective is k–Center, where the goal is to find k centers that minimize the maximum distance of a point to its closest center. It is well known that this can be approximated to within a factor of 2 [19, 12].

Guha *et al.* [16] gave an n^ϵ-space algorithm guaranteeing an $O(2^{O(1/\epsilon)})$-approximation to the k–Median problem on streaming data. Earlier, Charikar *et al.* [7] gave a streaming algorithm for the k–Center problem that gives a constant factor approximation using $O(k)$ space. In fact, the algorithm in [7] maintains at most k clusters at all times and modifies the clustering solution using a very restricted set of operations that do not allow an existing cluster to be broken up and its points distributed to different clusters. Recently, Charikar and Panigrahy [11] gave a streaming algorithm for the objective of finding k clusters so as to minimize the sum of cluster diameters. Their streaming algorithm requires $O(k)$ space and achieves a constant factor approximation but also increases the number of centers by a constant factor; it works under the same set of restricted cluster update operations as in [7].

Intuitively, a streaming algorithm for clustering that maintains a small amount of state must maintain a succinct representation of the clustering solution (usually just the set of cluster centers plus some additional information). Addition-

*Email: moses@cs.princeton.edu. Research supported by NSF ITR grant CCR-0205594 and DOE Early Career Principal Investigator award DE-FG02-02ER25540.

†Email: loc@cs.stanford.edu. Research supported by NSF Grant IIS-0118173 and an ARCS scholarship.

‡Email: rinap@cisco.com.

STOC'03, June 9–11, 2003, San Diego, California, USA.

ally, it must maintain an implicit *lower bound* on the optimal solution for the sequence seen so far. Thus it appears that any clustering objective function that admits such a clustering algorithm must have succinct lower bound certificates on the cost of the optimal solution. For the k–Center objective (minimize maximum cluster radius), such a succinct certificate follows directly from the analysis of the offline k–Center problem. The existence of such a certificate was the basis for the streaming algorithm for clustering with the k–Center objective in [7]. In the case of the sum of diameters objective [11], the succinct lower bound certificate is non-trivial; however, a lower bound certificate consisting of $O(k)$ carefully chosen points can be obtained from a careful analysis of the primal and dual LP formulations for this problem.

While the streaming algorithms for k–Center and sum of cluster diameters use only $O(k)$ space and yield constant factor approximations, the currently best known streaming algorithm for k–Median requires $O(n^\epsilon)$ space and produces a $2^{O(1/\epsilon)}$ approximation [16]. Our research was motivated by the following question, left open by the work of Guha *et al.* : *Is it possible to devise a streaming algorithm for the k–Median problem that uses only $O(k\,poly\log(n))$ space and produces a constant (or even logarithmic) approximation factor ?*

Our Results: Our main result solves this open problem from [16]. We give a streaming algorithm for the k–Median problem that uses $O(k\,poly\log(n))$ space and produces a constant factor approximation; the algorithm is randomized and works with high probability. Roughly speaking, the algorithm of Guha *et al.* performs the clustering at several levels: at level 1, blocks of the input of size n^ϵ are clustered. These cluster centers from level i form the input to level $i+1$. Since n^ϵ points can be processed simultaneously, the algorithm needs $1/\epsilon$ levels to process the entire input. The analysis in [16] shows that the approximation factor increases by a constant factor in going from level i to $i+1$; therefore the algorithm gives an approximation guarantee of $2^{O(1/\epsilon)}$. Our algorithm can be summarized as follows: We obtain a clustering solution with $k\log n$ medians which is only a constant factor worse than the optimal (with k medians). The final medians are obtained by solving a k–Median problem on these $O(k\log n)$ medians (appropriately weighted). The crux of our approach is maintaining a constant-factor-approximate solution using $k\log n$ medians and using $O(k\,poly\log n)$ space in the streaming model. In this respect, our approach has some parallels with the recent work of Gilbert *et al.* [14] on approximate histogram maintenance in the streaming model. In order to obtain a good approximation to the best B-bucket histogram, they instead maintain a solution with a much larger number of buckets; this solution is then used to obtain a good B-bucket solution in the end.

Our algorithm operates in phases; we also have a lower bound on the optimal cost, which is updated from phase to phase. Each phase uses the recent online facility location algorithm by Meyerson [30]. (Facility location algorithms have been used to solve k–Median by several previous works, e.g. the work of Jain and Vazirani [23]). Though Meyerson's algorithm only guarantees an $O(\log n)$ approximation for online facility location[1], we are able to exploit it to maintain a constant-factor approximation for k–Median by increasing the number of centers by an $O(\log n)$ factor. Analysis similar to that used in [16] shows that in updating our solution from phase to phase, we face a potential multiplicative increase in the approximation ratio, but we show how to overcome this problem and guarantee a constant-factor approximation.

Next, we study the asymmetric k–Median problem. Given an upper bound Δ on the ratio of the maximum distance to the minimum distance, we give a streaming algorithm which makes $O(\log\Delta + \log n)$ passes and uses space $O(k^2(\log\Delta + \log n)/\epsilon^2)$ to produce a $(1+\epsilon)$-approximate solution that uses $O(k(\log\Delta+\log n)/\epsilon)$ centers. We note that an $O(\log n)$ increase in the number of centers is inevitable for this problem [26].

We also study outlier formulations of clustering problems in the streaming model: For a given parameter δ, find a solution that clusters at least $(1-\delta)$ fraction of the points and optimizes the objective function on these chosen points. (In other words, we are allowed to exclude at most δ fraction of the points, called outliers, from the clustering). Charikar *et al.* [10] recently introduced these outlier formulations and gave constant-factor algorithms for the outlier versions of the k–Center and k–Median problems. We obtain $O(1)$-approximations for these problems in the streaming model using $O(k\log n)$ space while increasing the fraction of outliers slightly (a bicriterion guarantee). We note that for the k–Median problem with outliers, even in the offline setting, it is not known how to obtain a solution with a constant factor approximation while maintaining the same fraction of outliers as the optimal solution. Our algorithms for these problems are extremely simple and very efficient when viewed as offline algorithms. We show that for both of these problems, solving an outlier clustering problem on a random sample of size $O(k\log n)$ gives a good solution for the entire data set. In either case, we obtain an implicit representation for the solution in the form of k cluster centers and a distance threshold; the points that have no cluster center within the distance threshold are considered outliers, and the rest are assigned to their closest centers.

Related Work: Mettu and Plaxton [28] considered a version of the k–Median problem they called the "Online Median Problem;" here the goal was to order the set of points in such a way that for any k, the first k points when treated as a k–Median solution were a good approximation to the best k–Median solution.

Several previous papers have developed time-efficient algorithms for the k–Median problem. For example, Indyk [20, 21] and Thorup [33] gave efficient algorithms for k–Median based on sampling. Meyerson *et al.* [31] devised sampling-based algorithms for k–Median that run in time $O(k^2\log(k)/\epsilon)$ (independent of n), assuming each cluster is of size at least $\epsilon n/k$, or that some fraction of the points can be treated as outliers. Their results are related to those presented here but use different algorithmic techniques and assume a different input model. In particular, they assume random access to the data points, which is not available in a stream, and the goal is to avoid all running time dependence on n. Mishra *et al.* [32] gave a sampling-based algorithm that, with probability at least $1-\delta$, produces a k–Median clustering of cost at most $\beta\textsc{opt}+\epsilon$, where $\textsc{opt}$ is the optimum k–Median cost, β is $O(1)$, and ϵ is an input parameter. The running time depends on $(M/\epsilon)^2$,

[1]The algorithm gives an expected $O(\log n)$-approximation, and, if inputs arrive in random rather than possibly adversarial order, it gives an expected constant-factor approximation.

where M is the diameter of the data set, and, in the case of a general metric space, on $\log(n/\delta)$. Mettu and Plaxton [29] gave an algorithm based on repeated sampling that, with high probability, produces an $O(1)$-approximate solution to k–Median, as long as the range of distances is bounded above by $2^{O(n/\log(n/k))}$. The algorithm runs in $O(n(k+\log n)+k^2\log^2 n)$ time, which is $O(nk)$ when k is $\Omega(\log n)$ and $O(n/log^2 n)$. They also showed that when the range of distances is large enough, no $o(nk)$-time, randomized algorithm can guarantee a constant approximation to k–Median with nonnegligible (e.g., above 0.01) probability.

Our results on outlier versions of clustering problems are closely related to the work of Alon *et al.* [2] on testing clustering. They considered the problem of distinguishing between the case that a set of points can be divided into k clusters with maximum radius b, and the case that an ϵ fraction of the points must be removed before the set can be clustered into k clusters of maximum radius b', where $b' > b$. They showed how to distinguish between the cases using a small random sample, and present a detailed analysis of the case when the points are in $\Re^d$.

Lin and Vitter [26] gave LP-based algorithms for the k–Median problem with arbitrary distances. Their filtering technique gets a $(1+\epsilon)$ approximation by using $(1+\frac{1}{\epsilon})(\ln n+1)k$ centers. Young [34] improved these results by giving a greedy algorithm which produces a $(1+\epsilon)$ approximation using $k\ln(n+n/\epsilon)$ centers.

2. K–MEDIAN ON STREAMS

Given a set X of n points from some metric space, an integer k, and k members $c_1,\dots,c_k$ of the metric space, we will say that the k–Median *cost* of using $c_1,\dots,c_k$ as medians for X (or, simply, the k–Median cost of $c_1,\dots,c_k$ on X) is $\sum_{x\in X}\min_{1\le i\le k}\{\mathrm{dist}(x,c_i)\}$. That is, the cost of a set of medians is the value of the k–Median objective function when these medians are used.

In this section we will introduce **PLS**, a k–Median algorithm for streams which runs in polylogarithmic space. If X is the input stream of n points, k is the desired number of medians, and C is the cost of the lowest-cost k–Median solution for X, **PLS** finds k medians from X whose cost is at most $O(1)$ times C. **PLS** requires a subroutine **ONLINE-FL** (to be described shortly) that, given a point stream, finds a set of medians for an initial segment of that stream. As mentioned before, **PLS** operates in phases, during each of which a further segment of the input stream is consumed. In each phase i, **PLS** invokes **ONLINE-FL** on a modified version X_i of the original input stream X, and calculates, for each median m found by **ONLINE-FL** for X_i, the number of points $w(m)$ (called the "weight" of median m) from X_i assigned to m. Each median found during phase i is then associated with its weight, and the weighted set of medians is stored as the "solution from phase i." For every i, the stream X_{i+1} input to phase $i+1$ is the solution from phase i, concatenated with the remaining unread points from X; in other words, X_{i+1} is just X, with the medians found in phase i replacing the initial segment of X that has been read before phase $i+1$. The first segment of X_{i+1} (the solution from phase i) consists of points with possibly non-unit weights; the second part has unit weights.

The subroutine **ONLINE-FL** that is invoked in each phase of **PLS** is Meyerson's online facility location algorithm [30]. We give a brief review of Meyerson's algorithm and then recast the guarantee in a form that will be convenient for the proofs in the rest of this section.

ONLINE-FL (data stream X, facility cost f)

1. Make one pass over X performing the following steps for each $x \in X$:
2. Let δ be the distance of the current point x to the closest already-open facility.
3. With probability δ/f (or probability 1 if $\delta/f > 1$), open a new facility at x. Otherwise, the new point is assigned to the closest already-open facility.

Recall that we will apply **ONLINE-FL** during each phase to a weighted point set. In processing a point x of weight w, the probability of building a facility at x in Step 3 of the algorithm is $\min(w\delta/f, 1)$.

Consider an instance of k–Median and let OPT be the cost of the optimal solution for this instance. Let L be a lower bound on OPT. (We will later explain how this lower bound is obtained). We will run the facility location algorithm with facility costs $f = \frac{L}{k(1+\log n)}$. The proof of the following lemma follows from the proof technique in [30].

LEMMA 1. *The expected number of medians in the solution produced by* **ONLINE-FL***, is at most* $k(1+\log n)(1+4OPT/L)$*. The expected cost is at most* $L+4OPT$*.*

PROOF. Let $c_1^*, c_2^*,\dots,c_k^*$ be the medians in the optimal solution. Let d_p^* denote the distance from point p to the nearest median in the optimum solution. Let C_i^* denote the set of points assigned to median c_i^*, and let $A_i^* = \sum_{p\in C_i^*}\mathrm{dist}(p,c_i^*)$. Define a_i^* to be $A_i^*/|C_i^*|$. Focus on some optimum cluster C_i^*. For $j=1,2,\dots,\log n$, let S_j be the set of points p in C_i^* for which $2^{j-1}a_i^* < d_p^* \le 2^j a_i^*$ from the median c_i^*. (Note that S_j is empty for $j > \log n$). Now, consider the points in S_j. The expected service cost (sum of distances of points to their medians) paid before a median is opened at some point in S_j is at most f. Any subsequent point $p \in S_j$ has distance at most $3d_p^*$ to this median. The service cost paid for this subsequent point is bounded by $3d_p^*$. On the other hand, the probability that a median is opened at p is at most $\frac{3d_p^*}{f}$.

Now consider the set of points within a_i^* of the optimum median c_i^*. The expected (service) cost paid before a median is opened in this set is at most f. After a median is opened within this set, the distance of a subsequent point p from its nearest median is at most $a_i^* + d_p^*$. The service cost paid for this subsequent point is bounded by $a_i^* + d_p^*$. On the other hand, the probability that a median is opened at p is at most $(a_i^* + d_p^*)/f$.

We will bound separately the expected number of medians opened by the algorithm, and the expected (service) cost paid by the algorithm. The expected number of medians opened by the algorithm (amongst points in C_i^*) is bounded by $1+\log n+\frac{1}{f}\left(\sum_{p\in C_i^*}(a_i^*+3d_p^*)\right) \le 1+\log n+\frac{4A_i^*}{f}$. Summing this over all clusters $C_i^*, i=1,\dots,k$, the total number of medians opened by the algorithm is bounded by $k(1+\log n)(1+4OPT/L)$.

Now let us bound the expected service cost paid by the algorithm. The expected service cost paid for points in C_i* is bounded by $f(1+\log n)+3\sum_{p\in C_i^*}(d_p^*+a_i^*) \le \frac{L}{k} + 4A_i^*$. Summing up over all clusters, the expected service cost paid by the algorithm is at most $L+4OPT$. □

The following corollary of Lemma 1 follows by a simple application of the Markov inequality and the union bound.

COROLLARY 1. *With probability at least* $1/2$*, algorithm* **ONLINE-FL** *produces a solution of cost at most* $4(L + 4OPT)$ *and using at most* $4k(1+\log n)(1+\frac{4OPT}{L})$ *medians.*

2.1 Phases and phase transitions

We now present the details of the algorithm for k–Median on streams. The algorithm and other subroutines invoked by it mark points in the stream "read" as they are processed. When we change phases, the algorithm might see a point, but not mark it as "read". We use the notion of "read" points to obtain a bound on the number of phases by ensuring that every phase makes progress, i.e. marks at least one new point as "read". Let p denote the number of phases. For ease of exposition, we initially describe a simpler (although slightly incorrect) algorithm and analyze it assuming that $p \leq n$. In the next section, we will modify the algorithm to ensure that the number of phases is at most n. The algorithm **PLS** is as follows, where β is a constant to be explained later, and k is assumed to be less than $n = |X|$.

PLS(point stream X, integers k and n)

1. $L_1 \leftarrow$ **SET-LB**$(X)/\beta$;
2. $i \leftarrow 1$; $X_1 \leftarrow X$;
3. while there are unread points in X:
   (a) $M_i \leftarrow$ **PARA-CLUSTER**(L_i, X_i, k, n);
   (b) $X_{i+1} \leftarrow M_i||(X - R_i)$;
   (c) $L_{i+1} = \beta L_i$;
   (d) $i \leftarrow i + 1$;
4. Return M_{i-1}, the medians given by the most recent invocation of **PARA-CLUSTER**.

The first call made by **PLS** sets a lower bound using the following procedure **SET-LB**:

SET-LB(point stream X, integer k)

1. Without marking any points of X as "read," let d equal the distance between the closest pair of the first $k+1$ members of X in order;
2. return d;

In step 3a of **PLS**, **PARA-CLUSTER** returns M_i, a set of (weighted) medians found for the part of X_i that it read. In step 3b, $X - R_i$ (to be defined more formally later) is the portion of X that is still unread at the end of phase i, and $M_i||(X - R_i)$ is the concatenation of M_i and $(X - R_i)$. The description of **PARA-CLUSTER** follows. In this description, β and γ are constants that will be chosen to satisfy the following condition:

$$\gamma + 4(1 + 4(\beta + \gamma)) \leq \gamma\beta \qquad (1)$$

PARA-CLUSTER(point stream S, lower bound L_i, integers k and n)

1. Run $2\log n$ parallel invocations of **ONLINE-FL** with $f = L_i/(k(1 + \log n))$ on the input S.
2. Each invocation is run as long as the number of medians does not exceed $4k(1+\log n)(1+4(\gamma+\beta))$ and the cost of the solution (on the modified input) does not exceed $4L_i(1 + 4(\gamma + \beta))$. If either of these conditions is violated by a particular invocation, we stop running this invocation.
3. Continue until all invocations have been stopped.
4. When the last invocation finishes, mark as "read" all points seen by this invocation, except for the last (i.e., the point that would cause the median limit or the cost limit to be exceeded); return the medians found by this invocation.

The ith phase of **PLS** consists of the ith call to the routine **PARA-CLUSTER**, which consumes an initial segment of the modified stream X_i passed to it as a parameter. The result of this phase is that the members of X_i that are seen during this ith call to **PARA-CLUSTER** (except for the last point to be seen) are marked as "read" so that the unread points can be appended untouched to the set of medians found by **PARA-CLUSTER** for X_i.

We will next prove that the cost of the medians returned by **PLS** is at most a constant times the cost of the best k–Median solution for X, and that there are at most n phases. The following notation will be useful in these proofs.

DEFINITION 1. *If k is a positive integer, and S_1 and S_2 are subsets of a common metric space with $|S_1| \geq k$ and $|S_2| \geq k$, let $D(S_1, S_2, k)$ be the minimum, over all $T \subseteq S_2$ with $|T| = k$, of $\sum_{x \in S_1} \min_{t \in T} dist(x, t)$. $D(S_1, S_2, k)$ is the cost of the best k medians for S_1, if the medians are allowed to be chosen from S_2.*

DEFINITION 2. *For $1 \leq i \leq p$, let R_i denote the subset of X that is marked as "read" during phases 1 through i. Let S_i denote the subset of X that is seen by the algorithm during phases 1 through i. Note that $R_i \subseteq S_i$, that in phases 1 through $p-1$, S_i contains only one more point than R_i (the point seen, but returned unread, by the latest-finishing invocation of* **ONLINE-FL***), and that in phase p $R_i = S_i$.*

DEFINITION 3. *If $1 \leq i < p$ let $OPT_i = D(S_i, X, k)$. (Recall that S_i is the subset of X that is seen in phases 1 through i.) Note that the solution that attains the cost OPT_i on S_i may include medians not in S_i.*

DEFINITION 4. *Define $A_1 = 0$, and let A_i denote the cost of the medians M_{i-1} on R_{i-1}, the points read (i.e., marked as "read") before phase i.*

FACT 1. *Each new point marked "read" by* **PLS** *adds at most f (the current value of facility cost when the point is read) to the solution cost.*

FACT 2. *The value returned by* **SET-LB** *to* **PLS** *is a lower bound on $D(X, X, k)$.*

PROOF. Note that Fact 1 implies that the first $k+1$ points are guaranteed to be read in the first phase. □

LEMMA 2. *Let $P(i)$ denote the event that $A_i \leq \gamma L_i$, and let $Q(i)$ denote the event that $L_{i+1} = \beta L_i \leq OPT_i$. For appropriate choice of β and γ, with probability $1 - \frac{1}{poly(n)}$, for all $1 \leq i < p$, $P(i)$ and $Q(i)$ hold.*

PROOF. First we will show $P(1)$, $P(2)$, and $Q(1)$. We will then show that if $P(i')$ holds for all $i' \leq i$ and $Q(i')$ holds for all $i' < i$, then $P(i+1)$ holds, and with probability at least $1 - 1/(n^2)$ $Q(i)$ holds as well. Since p, the number of phases of **PLS**, is at most n, it will follow that with probability at least $1 - 1/n$ $P(p-1)$ and $Q(p-1)$ both hold.

$P(1)$ holds trivially as long as $\gamma \geq 0$, because $A_1 = 0$. $P(2)$ holds since $4(1+4(\gamma+\beta)) \leq \gamma\beta$ (this follows from the condition (1)), because A_2 is guaranteed not to exceed $4(1+4(\gamma+\beta))L_1$, and $L_2 = \beta L_1$. Finally, $Q(1)$ is also true, because L_1 is set to $1/\beta$ times a lower bound on $D(S_1, X, k)$.

We will next prove that, with high probability, P and Q hold for all phases except the last. Let $i < p$ be such that $\forall i' \leq i$, $P(i')$, and $\forall i' < i$, $Q(i')$. We will show that with high probability $Q(i)$ holds, and that, if $i+1 \leq p-1$, $P(i+1)$ holds. Suppose for contradiction that $OPT_i < \beta L_i$. Let $OPT_i' = D(M_{i-1} \cup (S_i - S_{i-1}), X, k)$, that is, the cost of the k members of X whose cost on the modified input seen during phase i is optimal. Note also that the solution that attains a cost of OPT_i on $M_{i-1} \cup (S_i - S_{i-1})$ could use medians outside the set of points seen in phases 1 through i. Then $OPT_i' \leq A_i + OPT_i$ (the proof is similar to the proof of Theorem 2.3 in [16], and so $OPT_i' \leq (\beta+\gamma)L_i$. Therefore, the modified instance seen in phase i must have a feasible solution costing $(\beta+\gamma)L_i$, and so, by Corollary 1, each invocation has a probability of at least 1/2 of *reading* the last member of S_i before overrunning the limits on cost or number of medians. However, since all invocations stop without reading this last point, either we have a contradiction (and so $OPT_i \geq \beta L_i$) or else a probability-$1/n^2$ event has occurred.

We also need to ensure that $A_{i+1} \leq \gamma L_{i+1}$ if $i+1 \leq p-1$. Let C_i denote the cost of M_i on $M_{i-1} \cup (R_i - R_{i-1})$, the modified input read during phase i. By definition of the algorithm, $C_i \leq 4(1+4(\beta+\gamma))L_i$. Note that the cost of M_{i-1} with respect to R_{i-1}, the points marked as read before phase i, is $A_i \leq \gamma L_i$. Then A_{i+1}, the cost of the solution that the algorithm has at the *end* of phase i (measured with respect to the original input points R_i), is at most $A_i + C_i$. However, since $\gamma + 4(1+4(\beta+\gamma)) \leq \gamma\beta$ (condition (1)), it follows that $A_i + C_i \leq \gamma L_{i+1}$, i.e., , $P(i+1)$ holds.

$Q(p-1)$ holds unless for some phase $i < p$ $Q(i)$ fails. We have seen that for each phase i before the last, if $Q(i-1)$ holds, this failure has probability at most $1/n^2$. In the next section, we will show there are at most n phases (by modifying the algorithm slightly). Since there are at most n phases, the probability that $Q(p-1)$ fails is at most $1/n$. □

If $Q(p-1)$ holds, then $L_{p-1} \leq OPT_{p-1}/\beta$. OPT_{p-1} is certainly no more than OPT_p, and so $L_p = \beta L_{p-1} \leq OPT_p$. Since the last phase finds a solution costing at most $4(1+4(\gamma+\beta))L_p$, the medians produced by phase p cost at most $4(1+4(\gamma+\beta))OPT_p$.

2.2 Bounding the number of phases

The analysis in the previous section assumes that there are at most n phases, but as the algorithm is presented, several phase changes may occur without the consumption of new input, and the number of phases could therefore be much higher than n. Next, we will see how to ensure that each phase marks at least one new point "read."

Let x denote the point in the data stream that is seen, but not marked "read," in phase i. We want to ensure that x *is* marked read in phase $i+1$. Recall that $OPT_i = D(S_i, X, k)$ is the cost of the optimal solution on $S_i = R_i \cup \{x\}$. We will compute an improved lower bound L_{i+1} for OPT_i so as to ensure that $OPT_i \leq \beta L_{i+1}$. It will follow that, at the termination of phase $i+1$, with very high probability the cost of the optimal solution on the points seen so far exceeds βL_{i+1}, and phase $i+1$ marks x as "read" before terminating.

We compute an approximate k–Median solution to the modified instance consisting of M_i (the weighted medians from phase i) and x. In order to do this, we use a c-approximation algorithm for k–Median. Let APX_i denote the value of the c-approximate solution obtained. We set L_{i+1} as follows:

$$\begin{aligned} L'_{i+1} &= \frac{APX_i}{2c(1+\gamma)} \\ L_{i+1} &= \max(\beta L_i, L'_{i+1}) \end{aligned}$$

In the previous section, we proved that the invariant $A_{i+1} \leq \gamma L_{i+1}$ is maintained by the algorithm. Note that the new procedure for setting L_{i+1} does not violate this invariant. Furthermore, examining the proof in the previous section, it is easy to see that the algorithm maintains the stronger invariant $A_{i+1} \leq \beta\gamma L_i$. We will also need the following condition on β and γ:

$$\beta \geq 2c(1+\gamma) + \gamma \tag{2}$$

Note that this does not contradict the condition (1) we have already imposed on β and γ. We first prove that L_{i+1} is a valid lower bound on the optimal cost.

Lemma 3. $L_{i+1} \leq OPT_i$.

Proof. We know that $OPT_i \geq \beta L_i$. We need to prove that $OPT_i \geq L'_{i+1}$. Recall that OPT_i is the value of the optimal k–Median solution on $R_i \cup \{x\}$. (This could use medians outside $R_i \cup \{x\}$). Also recall that A_{i+1} is the cost of assigning the points in R_i to the medians in M_i. This implies that there is a k–Median solution for the weighted points in M_i together with x, with cost at most $OPT_i + A_{i+1} \leq OPT_i + \beta\gamma L_i$. (Again, this could use medians outside $M_i \cup \{x\}$). Further, we can transform this solution to a solution that uses medians only in $M_i \cup \{x\}$, with cost at most $2(OPT_i + \beta\gamma L_i)$. Since we use a c-approximation algorithm to compute a k–Median solution on this instance, $APX_i \leq 2c(OPT_i + \beta\gamma L_i)$.

$$\begin{aligned} L'_{i+1} &= \frac{APX_i}{2c(1+\gamma)} \leq \frac{2c(OPT_i + \beta\gamma L_i)}{2c(1+\gamma)} \\ &= \frac{1}{1+\gamma} OPT_i + \frac{\gamma}{1+\gamma}\beta L_i \\ &\leq OPT_i \end{aligned}$$

The last inequality follows since $\beta L_i \leq OPT_i$. □

Next, we prove that that OPT_i is upper bounded by βL_{i+1} (which ensures progress in each phase). First, we state and prove a bound we will need in the proof.

Claim 1. $APX_i \geq OPT_i - A_{i+1}$.

Proof. The statement of the claim is equivalent to $APX_i + A_{i+1} \geq OPT_i$. Note that APX_i is the cost of a k–Median solution on the weighted medians in M_i and the point x. A_{i+1} is the cost of assigning the points in R_i to the medians in M_i. By combining these two solutions, we get a feasible k–Median solution for the points in $R_i \cup \{x\}$ with cost at most $APX_i + A_{i+1}$. Since OPT_i is the cost of the optimal solution for this instance, it follows that $APX_i + A_{i+1} \geq OPT_i$. □

Lemma 4. $OPT_i \leq \beta L_{i+1}$.

PROOF.

$$\begin{aligned} L'_{i+1} &= \frac{APX_i}{2c(1+\gamma)} \geq \frac{OPT_i - A_{i+1}}{2c(1+\gamma)} \\ 2c(1+\gamma)L'_{i+1} + A_{i+1} &\geq OPT_i \\ (2c(1+\gamma)+\gamma)L_{i+1} &\geq OPT_i \\ \beta L_{i+1} &\geq OPT_i \end{aligned}$$

The last step follows because β and γ are chosen so that (2) holds. □

2.3 The final medians

In order to produce the final medians, we simply solve a k–Median problem on the final set of $O(k \log n)$ medians, treating the medians as points weighted by the number of points assigned to them. Let OPT be the value of the optimal solution. Suppose we end up with a solution with $k \log n$ centers S with cost at most αOPT. (α is a constant). Then there exists a solution for the weighted instance on S with k medians amongst the points of S with cost at most $2(1+\alpha)OPT$. (The proof follows from Theorem 2.3 in [16]). If we use a c-approximation algorithm for this k–Median instance, we get a solution of cost at most $2c(1+\alpha)OPT$ for the weighted instance on S. In going from the points in S to the original points, we must pay an additional αOPT, yielding a solution of cost at most $(\alpha + 2c(1+\alpha))OPT$ for the original set of points.

THEOREM 1. *The streaming algorithm for k–Median produces an $O(1)$ approximation, in one pass, with probability $1 - \frac{1}{poly(n)}$; the space required is at most the space to represent $O(k \log^2 n)$ points from the stream.*

3. ALGORITHMS FOR ASYMMETRIC K–MEDIAN

Next we present algorithms for solving k–Median on sets with asymmetric distance functions. Here, distances are directed, so that $\text{dist}(x, y)$ may not equal $\text{dist}(y, x)$. However, if X is the point set, $\forall\ x,\ y,\ z \in X$ $\text{dist}(x, x) = 0$, and $\text{dist}(x, z) \leq \text{dist}(x, y) + \text{dist}(y, z)$ (that is, the distances obey a directed triangle inequality). Initially, we will assume that all inter-point distances are finite and positive; later we will show how to modify the algorithm in case some inter-point distances are zero. The algorithms are based on the idea of growing ever-larger clusters around medians until all points have been assigned. Initially, a small coverage radius r_1 will be set, and all points will be regarded as unassigned. If for some point x at least some minimum fraction of still-unassigned points can be assigned to x for a cost of at most r_1 apiece, x will be made a median, and its neighbors within r_1 will be marked as assigned, or "covered." We will use the terms "assigned" and "covered" interchangeably. Once there are no points with enough neighbors within r_1, the radius will be increased (doubled, say) to r_2, and we repeat the same procedure. We can show that every time we increase the radius, the number of points assigned by our algorithm is almost as many as the number assigned within the old coverage radius by the optimum k–Median solution.

Non-streaming Algorithm. Given n points with a possibly asymmetric distance function such that the ratio Δ of the largest distance d_{max} to the smallest inter-point (i.e., positive) distance d_{min} is known, and input parameters ϵ and k, the algorithm will produce a solution of at most $2k \log \Delta/\epsilon$ medians, whose cost is at most a constant times the optimum cost. The algorithm proceeds in at most $2 \log \Delta$ phases, with radius r_i for phase i, where $r_i = (1+\epsilon)r_{i-1}$. We will discuss later how to find a good value for r_1, but for now we will assume we have an $r_1 > 0$.

DEFINITION 5. *A point x is said to i–cover a point y if $dist(x, y) \leq r_i$.*

Initially, all points will be unassigned, and in each phase we will "cover" more points, that is, assign them to new medians. For every phase i (call the first phase 1), let U_i denote the number of points that are still unassigned after this phase; define $U_0 =$ n. In phase i we will consider each $x \in X$ in turn; if a point x i-covers at least $\epsilon U_{i-1}/k$ of the unassigned points, we will mark them as "covered" and make x a median. [2] After phase i, we will set $r_{i+1} = (1+\epsilon)r_i$, and then begin phase $i+1$. We will continue until all points are assigned.

LEMMA 5. *For every point x in the data set, let $C^*(x)$ denote the median to which x is assigned under the optimal solution. Let $U_i^* = |\{x | dist(x, C^*(x)) > r_i\}|$, that is, the number of points that cost more than r_i in the optimal solution. For every phase i, $U_i \leq U_i^* + \epsilon U_{i-1}$.*

The proof follows from the fact that if no point i-covers an ϵ/k fraction of what remained unassigned after the last phase, then in particular none of the optimum medians does.

Set $r_1 = d/\Delta$ for an arbitrary inter-point distance d. Observe that $r_1 \leq d_{min}$, and so $nr_1 \leq nd_{min}$. Suppose that $k \leq n/2$. (In fact $k \leq \min\{n/\log n, n/\log \Delta\} \leq n/2$, or else the trivial solution with n medians is feasible in our framework.)

THEOREM 2. *The above algorithm finds $O(k \log \Delta)$ medians, whose cost is at most a constant times the optimum k–Median cost.*

PROOF. Let OPT denote the cost of the optimum medians, and COST the cost of the medians found by the algorithm described above. Then $\sum_{i=1}^{m-1}(r_i - r_{i-1})U_i \leq$ COST $\leq \sum_{i=0}^{m-1}(r_{i+1} - r_i)U_i$, and OPT $\geq \sum_{i=1}^{m-1}(r_i - r_{i-1})U_i^* \geq \frac{1}{1+\epsilon}\sum_{i=1}^{m-1}(r_{i+1} - r_i)U_i - \frac{\epsilon}{1+\epsilon}\sum_{i=1}^{m-1}(r_{i+1} - r_i)U_{i-1}$. It follows that OPT $\geq$ COST$(1 - \epsilon(1+\epsilon)^2)/(1+\epsilon)$ $-\frac{1+\epsilon^2}{1+\epsilon}r_1 n$. Note that because OPT $\geq (n-k)d_{min} \geq nd_{min}/2$, we have $r_1 n \leq 2$OPT. Then COST $\leq \frac{3+\epsilon+2\epsilon^2}{1-\epsilon(1+\epsilon)^2}$OPT, so the algorithm gives an $O(1)$-approximation.

As r_1 is a lower bound on the smallest inter-point distance in X, the algorithm will terminate in at most $2\log_{1+\epsilon}\Delta$ rounds, since if r_i reaches $d\Delta$ it need not be expanded further. Therefore, $O(k \log \Delta)$ medians are produced. □

Streaming Algorithm. Next we will describe how to apply the above algorithm using less memory, in a small number ($O(\log \Delta)$) of passes. The new algorithm uses $O(k^2 \log \Delta)$ space. If memory is scarce, counting the number of new points i-covered by some $x \in X$ requires a pass over X. Therefore, when deciding whether to make x a median, we

[2] Note that each point need only be considered once per phase as a potential median.

will *estimate* the true i-coverage of x using a pre-drawn sample. We cannot guarantee that the condition from Lemma 5 holds, but we show that with high probability each round covers a large enough proportion of the points covered by the optimal algorithm.

We will set r_1 and U_0 as before, and choose medians in phases. Every phase i will use three passes over X. First, we will in one pass choose k/ϵ independent random samples from the set of points that are still not covered. We will make a second pass over the data set, examining each point and making it a median if its estimated coverage is high enough. Finally, in the third pass, we will find U_i, the number of points still not covered.[3] (In phase $i+1$, the k/ϵ samples will be taken from these U_i.) The algorithm follows (the linear passes over X are noted):

STREAM-AKM(data set X, integer k, $0 \leq \epsilon \leq 1$)

1. Set r_1.
2. $i \leftarrow 1$.
3. While $i \leq 2\log\Delta$, do:
   (a) Set $p \leftarrow 1$.
   (b) (First pass): Draw k/ϵ independent samples $S_{i,1}$, $\ldots$, $S_{i,k/\epsilon}$, each of $\frac{8k\log(nk\log\Delta)}{\epsilon}$ points drawn uniformly at random (with replacement) from the U_{i-1} unassigned points.
   (c) (Second pass): Exit this pass and go to Step 3d when the last $x \in X$ is examined, or $p > k/\epsilon$. For each x, if x i-covers at least $\log n$ (remaining) points in the pth sample ($S_{i,p}$), do:
      i. Make x the next median from the current ith phase. Call this median $m_{i,p}$.
      ii. $p \leftarrow p+1$.
      iii. Remove from $S_{i,p}$ all points that are covered by medians $m_{i,1}$, $\ldots$, $m_{i,p}$. If $S_{i,p}$ becomes empty after this removal, continue increasing p by 1 and removing covered points from $S_{i,p}$ until an $S_{i,p}$ is found that is non-empty after the removal step, or until $p > k/\epsilon$.
   (d) (Third pass): Either the point stream or the set of samples was exhausted. Perform another pass to identify points in X that are not covered by the medians $m_{i,1}$, $\ldots$, $m_{i,k/\epsilon}$.
   (e) $r_{i+1} \leftarrow (1+\epsilon)r_i$; $i \leftarrow i+1$.

The sample size and the condition for becoming a median are chosen so that Lemma 5 above holds with high probability. The condition for becoming a median is that a point have high enough coverage on the still-unassigned points from the current (pth) sample. Because of the size of the samples and the fact that they are drawn independently (so that earlier medians are not picked on the basis of how many points they cover from *later* samples), a large coverage on a sample implies (with high probability) a large coverage on the set of all unassigned points. Chernoff bounds guarantee that if, during a phase i, a point x covers many points left unassigned after the $(i-1)$th phase, it will with probability greater than $k(nk\log\Delta)^{-1}$ cover at least $\log n$ points on the current sample, and will therefore become a median. Therefore when phase i terminates, the probability is less than $(n\log\Delta)^{-1}$ that there will be more than ϵU_{i-1} unassigned points whose assignment distances under the optimum solution are r_i or less. As there are at most $\log\Delta$ phases, with probability at least $1-1/n$ Lemma 5 holds throughout the execution of the algorithm. At most $\frac{k}{\epsilon}\log\Delta/$ medians are produced.

[3]This counting can be done in a single pass, but note that, later, determining whether a specific point is covered can only be accomplished by a computation of the distance to each existing median.

Zero Distances. If many points have zero assignment cost under the optimal solution, the above algorithm may make assignments costing as much as $r_1 > 0$ even if there is a zero-cost solution. Running **ZP**, a preprocessing algorithm amounting to an execution of **STREAM-AKM** with a few modifications, solves the problem. **ZP** adds another $O(\log n)$ passes over X, and finds $O(\log n)$ additional medians.

The changes are as follows. Instead of setting an initial radius r_1 and then having $O(\log\Delta)$ phases, each with a new radius $r_i = (1+\epsilon)^{i-1}r_1$, we will use the same coverage radius $r_0 = 0$ in each phase, and we will have $\log_{1/\epsilon} n$ phases. The sample size will change to $8k\log(nk\log_{1/\epsilon} n)/\epsilon$, and i-covering will be changed to "0-covering": x 0-covers y if $\text{dist}(x,y) = 0$. We can dispense with Step 3e. **ZP** will conclude with the following clean-up step: if there is exactly one member of X that is still unassigned after all phases, make this point a median. **ZP** will make at most $(k/\epsilon)(\log_{1/\epsilon} n)+1$ medians.

Suppose that in the optimal solution with k medians, exactly αn members of X have an assignment distance of 0; call these the "cheap" points, and the rest "expensive." Let $U_0 = \alpha n$ (a quantity not known to **ZP**). We will start off with all members of X unassigned, and U_i will denote the number of cheap points that are still unassigned after phase i of **ZP**. For each phase i, the probablility is at least $1 - 1/(n\log_{1/\epsilon} n)$ that $U_i \leq \epsilon U_{i-1}$. Therefore, after all $\log_{1/\epsilon} n$ phases, with probability at least $1-1/n$, the number of unassigned cheap points is at most α. Note that if α is 1 (the optimal solution has zero cost), a single unassigned point may remain, but the clean-up step removes it. Hence, no cheap points remain after **ZP** finishes.

Let n' denote the number of members of X that remain unassigned after **ZP**. Note that all these points are expensive (i.e., have positive assignment distance under the optimal solution). If $n' = \delta n$, then $\alpha \leq 1-\delta$. If $\delta = 1$, we have found a zero-cost solution (albeit with $O(k\log n)$ medians), and we can return this solution without running the rest of the algorithm. Otherwise, we run the algorithm **STREAM-AKM** using all n members of X as feasible medians, but only trying to cover the n' unassigned points. We will return the $O(k\log n)$ medians found by **ZP** and the $O(k\log\Delta)$ medians found by **STREAM-AKM**. If $\delta \geq 1/2$, then at least half the points in the optimal solution have assignment cost at least d_{min}, and so $r_1 n \leq 2\text{OPT}$. In this case, the analysis, and the cost approximation factor, from the previous section still hold. Suppose instead that $\delta < 1/2$. Let OPT' denote the sum of assignment distances of the remaining n' points, under the optimal solution. Then $\text{OPT} \geq \text{OPT}' \geq n' d_{min} \geq r_1 n'$. In this case, the analysis from the previous section shows that the cost of the solution we find is at most $\frac{2+\epsilon+\epsilon^2}{1-\epsilon(1+\epsilon)^2}\text{OPT}$.

4. K–CENTER WITH OUTLIERS

In this section, we will present an algorithm for the k–

Center problem given a metric distance function. In general, the object of the k–Center problem is to find cluster centers, and associate points with these centers, so as to minimize the largest cluster radius. The radius of the jth cluster C_j is the distance from the center c_j to the farthest member of C_j. If the problem requirements are relaxed so that a small fraction of the points can be *excluded*, that is, not assigned to clusters, much cheaper solutions may become available. Clearly, as the number that may be excluded increases, the lowest-achievable cost cannot rise.

Suppose that for some set X, r is the smallest maximum radius that can be achieved if at most δ fraction of X can be excluded. We present a one-pass algorithm with space complexity $O(k \log n)$ that will (with high probability) find a solution that excludes only a bit more than a δ fraction and achieves a maximum radius that is at most a constant times r. The algorithm is based on the principle that a random sample taken from X is roughly as "clusterable" as X itself. That is, if a particular set of centers covers a $(1-\delta)$ fraction of X within radius r, these same centers should cover approximately the same fraction of a sample of X within the same radius. We now describe the model and algorithm more formally.

Let a solution to k–Center be denoted by a set $\{c_1, \ldots c_k; r\}$ where the c_i are centers and $r \geq 0$ is a radius. If $T = \{c_1, \ldots, c_k; r\}$ is a k–Center solution for a set X, say that $x \in X$ is *assigned* under T if $\exists c_i$ such that $\mathrm{dist}(x, c_i) \leq r$. If a point x is not assigned under T, we say T *excludes* x. Call T an (r, δ)-solution if at least a $(1-\delta)$ fraction of the points in X are assigned under T.

If T is an (r, δ)-solution for X, and S is a subset of X, then T can also be viewed as a solution for S. The radius will not change when T is reinterpreted as a solution for S, but, given this radius, T may exclude a different fraction of S than of X. Note that T can be a valid solution even if it contains centers that are not in S. A solution T' for S can similarly be interpreted as a solution on X.

Assume the existence of a k–Center algorithm $\mathbf{A}$ that takes a data set X, an integer k, and a $0 \leq \delta \leq 1$, and returns k centers. Assume that there is a constant $\alpha > 1$ such that, if r^* is the smallest radius for which X has an (r^*, δ)-solution, $\mathbf{A}$ returns an (r', δ)-solution for X, where $r' \leq \alpha r^*$.[4] Then we have the following algorithm:

OKC (data set X, integer k, $0 < \epsilon < 1$, $0 < \delta < 1$)

1. Take a sample S of s points, uniformly at random with replacement, from X
2. Let $T = \{c_1, \ldots, c_k; r\}$ be the output of $\mathbf{A}(S, (1+\epsilon)\delta)$.
3. Return T, interpreted as a solution on X.

T may exclude a different fraction of X than of S. As we shall see, however, with probability at least $1 - 2/n^2$, T will not exclude more than $(1+\epsilon)^2\delta$ fraction of X, and r will be at most α times the smallest possible radius for a solution that excludes at most δ fraction of X.

Lemma 6. *Let S be a sample of $s \geq 8\frac{\ln n}{(\delta\epsilon^2)}$ points taken uniformly at random from X. Let T be an (r, δ)-solution on X. Then, with probability at least $1 - 1/n^2$, T excludes at most a $(1+\epsilon)\delta$ fraction of S.*

Proof. By Chernoff bounds, the chance that more than a $(1+\epsilon)\delta$ fraction of S is from the δ fraction of X excluded by T, is at most $1 - 1/n^2$. $\square$

[4]Such an algorithm is given, with $\alpha = 3$, in [10].

Lemma 7. *Let S be a sample of $s \geq \frac{2(1+\epsilon)(4+k)\ln n}{\epsilon^2 \delta_s}$ points taken uniformly at random with replacement from X. Consider an (r, δ_s)-solution T on S that is an (r, δ')-solution on X. The probability that $\delta' \leq (1+\epsilon)\delta_s$ is at least $1 - 1/n^2$.*

Proof. There are fewer than $n^{2+k}(e/k)^k$ solutions with radius at most r that exclude more than $(1+\epsilon)\delta_s$ fraction of X. Chernoff bounds guarantee that the probability that a given such solution excludes only δ_s on S is less than $e^{-s\delta_s\epsilon^2/(2(1+\epsilon))}$. Then regardless of S, there is probability at most $1/n^2$ that an (r, δ_s)-solution for S must exclude $(1+\epsilon)\delta_s$ on X. $\square$

Theorem 3. *Let $T = \{c_1, \ldots, c_k; r\}$ be the solution produced by $\mathbf{OKC}(X, k, \epsilon, \delta)$. Suppose T is an (r', δ')-solution on X. With probability more than $1 - 2/n^2$, the following are both true:*

- $\delta' \leq (1+\epsilon)^2\delta$
- *If the optimal solution that excludes at most a δ fraction of X has radius r^*, then $r' \leq \alpha r^*$*

Proof. Suppose there is an (r, δ)-solution on X. By Lemma 6, with probability at least $1 - 1/n^2$ there will be a solution on S with radius r that excludes at most $(1+\epsilon)\delta$. In this case, the solution returned by $\mathbf{A}(S, (1+\epsilon)\delta)$ will be an $(\alpha r, (1+\epsilon)\delta)$-solution on S. By Lemma 7, with probability at least $1 - 1/n^2$ this solution will exclude at most $(1+\epsilon)^2\delta$ on X. The theorem follows. $\square$

5. K–MEDIAN WITH OUTLIERS

We give a similar algorithm for "k–Median with Outliers." Recall that in k–Median the objective is the sum (or average) of assignment distances. If a solution is allowed to exclude some outliers, the cost will be the average cost (distance to the closest median) of the points that are *not* excluded. We present a one-pass, $O(k \log n)$-space, bicriterion algorithm: if the best solution that excludes at most a δ fraction of the points has some average assignment distance r, this algorithm finds a solution that excludes only slightly more than a δ fraction and achieves an average assignment distance that is $O(r)$.

A solution T to k–Median for a data set X of n points will consist of a set $\{c_1, \ldots, c_k\}$ of medians. For all $x \in X$, let $C(x)$ denote the closest median to x (that is, c_i such that $\forall c_j \neq c_i$, $\mathrm{dist}(x, c_i) \leq \mathrm{dist}(x, c_j)$). For all $x \in X$, call $\mathrm{dist}(x, C(x))$ the *radius* of x under T. If T can exclude a δ fraction of X, it will of course exclude the δn members of highest radii. Every $x \in X$ that is not excluded will be *assigned*, and the radius of x will be called the *assignment distance* of x.[5] T will be called an (r, δ)-solution for X if it has average assignment distance r when it assigns a $(1-\delta)$-fraction of the points in X.

As before, for $S \subseteq X$, any solution on X can be interpreted as a solution on S, and vice versa. In this case, when we reinterpret a solution for X as a solution for S (or vice versa), we will only keep the medians (not necessarily the radius or fraction excluded) the same. Assume the existence of a k–Median algorithm $\mathbf{B}$ that takes a data set X, an integer k, and a $0 \leq \delta \leq 1$, and returns k medians. Assume

[5]The cost of the solution will then be the average assignment distance, since other radii are not counted.

constants $\gamma > 1$ and $\beta > 0$ such that if r^* is the smallest radius for which X has an (r^*, δ)-solution, **B** returns an (r', δ')-solution for X, where $r' \leq \gamma r^*$ and $\delta' \leq (1+\beta)\delta$.[6]

OKM (data set X, integer k, $0 < \epsilon < 1$, $0 < \delta < 1$)

1. Take a sample S of s points, uniformly at random with replacement, from X
2. Let $T = \{c_1, \ldots, c_k\}$ be the output of $\mathbf{B}(S, (1+\epsilon)\delta)$.
3. Return T, interpreted as a solution on X.

Lemma 8. *Let S be a sample of s points drawn uniformly at random with replacement from X.*

Let $s \geq \max\{\frac{4 \ln n}{(1+\rho)\ln(1+\rho)-\rho} \times \frac{(1-\delta)/\delta\epsilon}{1-(1+\epsilon/2)\delta}, \frac{2 \ln n}{\delta}\}$. Let T be an (r, δ)-solution on X. Then, with probability at least $1 - 1/n^2$, there exist $r' \leq (1+\rho)r$ and $\delta' \leq (1+\epsilon)\delta$ so that T is an (r', δ') solution on S.

Proof. We first use Chernoff bounds to upper-bound (with high probability) the maximum of the cheapest $(1+\epsilon)\delta$ assignment distances for T on S. Next we apply them again to limit the average assignment distance for T on S. □

Lemma 9. *Let S be a sample of s points taken uniformly at random with replacement from X. Assume that $s \geq \max\{\frac{2(k+4)(2+\epsilon)\ln n}{\epsilon\delta_s}, \frac{16(k+4)\ln 2n}{C\delta_s\epsilon}, \frac{8(k+4)\ln 2n}{\delta_s\epsilon}(\frac{1+\alpha}{\alpha})^2\}$, where $C = \frac{2(1-(1+\epsilon/2)\delta_s}{\epsilon\delta_s}$. Consider an (r_s, δ_s)-solution T on S. Then, with probability at least $1 - 1/n^2$, there exist $r' \leq (1+\alpha)r_s$ and $\delta' \leq (1+\epsilon)\delta_s$ such that T is an (r', δ')-solution on X.*

Proof. We first fix a specific T' that is an (r, δ)-solution on X, where $r > (1+\alpha)r_s$ and $\delta > (1+\epsilon)\delta_s$. In two steps, we show that with high probability there do not exist $r'_s \leq r_s$ and $\delta'_s \leq \delta_s$ such that T' is an (r'_s, δ'_s)-solution on S. In the first step, we show a (high probability) lower bound on the highest assignment distance under T' on S if T' can only exclude δ_s of S. In the second step we show a (high probability) lower bound on the average assignment cost for T' on S when only δ_s can be excluded. It follows that with probability at least $1 - n^{-(k+4)}$, T' either excludes more than δ_s or has an average radius higher than r_s. We conclude the proof by noting that $n^{(2+k)}e^k/k^k$ is an upper bound on the number of possible solutions T', and that the probability that a particular T' of the type considered can be an (r_s, δ_s)-solution on S is less than $e^{\ln(2)-(4+k)\ln n}$. The product of these numbers is less than n^{-2}, and so, regardless of S, the probability that an (r_s, δ_s)-solution on S would have average assignment cost more than $(1+\alpha)r_s$ on X, even if it could exclude $(1+\epsilon)\delta_s$, is at most n^{-2}. □

Theorem 4. *Let T be the solution found by* **OKM** *for input (X, k, ϵ, δ). Suppose that, over all solutions for X that exclude at most a δ fraction of the points, r^* is the lowest possible average assignment distance. Then, with probability more than $1 - 2/n^2$, T is a $(\gamma(1+\alpha)(1+\rho)r^*, (1+\beta)(1+\epsilon)^2\delta)$-solution for X.*

The proof is similar to that for Theorem 3.

6. REFERENCES

[1] M. Ajtai, T. S. Jayram, R. Kumar, and D. Sivakumar Approximate counting of inversions in a data stream. *Proc. 34th STOC*, 2002.

[2] N. Alon, S. Dar, M. Parnas and D. Ron. Testing of clustering. *Proc. 41st FOCS*, 2000.

[3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *JCSS* 58(1): 1999.

[4] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k–median and related problems. *Proc. STOC*, 1998.

[5] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k–median and facility location problems. *Proc. STOC*, 2001.

[6] Z. Bar-Yossef, S. R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. *Proc. SODA*, 2002.

[7] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *Proc. STOC*, 1997.

[8] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k–Median problems. *Proc. FOCS*, 1999.

[9] M. Charikar, S. Guha, E. Tardos, and D. Shmoys. A constant factor approximation algorithm for the k–Median problem. *Proc. STOC*, 1999.

[10] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. *Proc. SODA* 2001.

[11] M. Charikar and R. Panigrahy. Clustering to minimize the sum of cluster diameters. *Proc. STOC*, 2001.

[12] M. Dyer and A. M. Frieze. A simple heuristic for the p-center problem. *Operations Research Letters*, v. 3, 1985.

[13] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L1-difference algorithm for massive data streams. *Proc. FOCS*, 2000.

[14] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. Fast, small-space algorithms for approximate histogram maintenance. *Proc. STOC*, 2002.

[15] S. Guha, N. Koudas, and K. Shim. Data-streams and histograms. *Proc. STOC*, 2001.

[16] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. *Proc. FOCS*, 2000.

[17] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems, part ii: p-medians. *SIAM Journal of Appl. Math*, v. 37, 1979.

[18] M. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. In DIMACS series in Discrete Mathematics and Theoretical Computer Science, v. 50, 1999.

[19] D. S. Hochbaum and D. B. Shmoys. A best possible approximation algorithm for the k–Center problem. *Mathematics of Operations Research*, v. 10, 1985.

[20] P. Indyk. Sublinear time algorithms for metric space roblems. *Proc. STOC*, 1999.

[21] P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. *Proc. FOCS*, 1999.

[22] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation.

[6]Such an algorithm is given in [10]. For example, β could be 2 and γ could be 6.

Proc. FOCS, 2000.

[23] K. Jain and V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual scheme and Lagrangian relaxation. *Journal of the ACM*, v. 48, 2001.

[24] M. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Proc. SODA*, 1998.

[25] J.-H. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, v. 44, 1992.

[26] J.-H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. *Proc. STOC*, 1992.

[27] G. S. Manku, S. Rajagopalan, B. G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. *Proc. SIGMOD Conference*, 1999.

[28] R. Mettu and C. G. Plaxton. The online median problem. *Proc. FOCS*, 2000.

[29] R. Mettu and C. G. Plaxton. Optimal time bounds for approximate clustering. *Proc. UAI*, 2002.

[30] A. Meyerson. Online facility location. *Proc. FOCS*, 2001.

[31] A. Meyerson, L. O'Callaghan and S. Plotkin. Approximating k–Median: A sampling-based approach. Unpublished manuscript, 2001.

[32] N. Mishra, D. Oblinger and L. Pitt. Sublinear time approximate clustering. *Proc. SODA*, 2001.

[33] M. Thorup. Quick k-median, k-center, and facility location for sparse graphs. *Proc. ICALP*, 2001.

[34] N. E. Young. K-medians, facility location, and the Chernoff-Wald bound. In *Proc. SODA*, 2000.